GEOMETRIC CONSTRAINTS SOLVING: SOME TRACKS

Dominique Michelucci, Sebti Foufou, Loic Lamarque *
LE2I, UMR CNRS 5158, Univ. de Bourgogne, BP. 47870, 21078 Dijon, France

Pascal Schreck
LSITT, Université Louis Pasteur, Strasbourg, France†

Abstract

This paper presents some important issues and potential research tracks for Geometric Constraint Solving: the use of the simplicial Bernstein base to reduce the wrapping effect in interval methods, the computation of the dimension of the solution set with methods used to measure the dimension of fractals, the pitfalls of graph based decomposition methods, the alternative provided by linear algebra, the witness configuration method, the use of randomized provers to detect dependences between constraints, the study of incidence constraints, the search for intrinsic (coordinate-free) formulations and the need for formal specifications.

CR Categories: J.6 [Computer Applications]: Computer Aided Engineering—CAD-CAM; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

Keywords: Geometric Constraints Solving, decomposition, witness configuration, Bernstein base, incidence constraint, randomized prover, rigidity theory, projective geometry

1 Geometric constraints solving

This article intents to present some essential issues for Geometric Constraints Solving (GCS) and potential tracks for future research. For the sake of conciseness and homogeneity, it focuses on problems related to the resolution, the decomposition, and the formulation of geometric constraints.

Today, all geometric modellers in CAD-CAM (Computer Aided Design, Computer Aided Manufacturing) provide some Geometric Constraints Solver. The latter enables designers and engineers to describe geometric entities (points, lines, planes, curves, surfaces) by specification of constraints: distances, angles, incidences, tangences between geometric entities. Constraints reduce to a system of (typically algebraic) equations. Typically, an interactive 2D or 3D editor permits the user to enter a so called approximate "sketch", and to specify geometric constraints (sometimes some constraints are automatically guessed by the software). The solver must correct the sketch, to make it satisfy the constraints.

Usually, the solver first performs a qualitative study of the constraints system to detect under-, well- and over-constrained parts; when the system is correct, *i.e.* well-constrained, it is further decomposed into irreducible well-constrained subparts easier to solve and assemble. This qualitative study is mainly a Degree of Freedom (DoF) analysis. It is typically performed on some kind of graphs [Owen 1991; Hoffmann et al. 2001; Gao and Zhang 2003; Hendrickson 1992; Ait-Aoudia et al. 1993; Lamure and Michelucci 1998]. This article presents the pitfalls of graph based approaches, and suggests an alternative method. After this qualitative study, if it is successful, irreducible subsystems are solved, either with some formula in the simplest case (*e.g.* to compute the intersection between two coplanar lines, or the third side length of a triangle knowing two other side lengths and an angle, etc), or with some numerical method, *e.g.* a Newton-Raphson or an homotopy, which typically uses the sketch as a starting point for iterations, or with interval methods which can find all real solutions and enclose them in guaranteed boxes (a box is a vector of intervals). Computer algebra is not practicable because of the size of some irreducible systems, and it is not used by nowadays' CAD-CAM modelers. In this article, we will show that in some cases using computer algebra is possible and relevant.

Depending on the context, either users expect only one solution, the "closest" one to an interactively provided sketch; or they expect the solver to give all real roots, and interval methods are especially interesting in this case. For instance, in Robotics, problematic configurations of flexible mechanisms are solutions of a set of geometric constraints: engineers want to know all problematic situations or a guarantee that there is none.

The paper is organized as follow: Section 2 discusses GCS using interval arithmetic and Bernstein bases. Problems related to the decomposition of geometric constraints systems (degree of freedom, scaling, homography, pitfalls of graph based methods, etc.) are discussed in Section 3. This section also provides some ideas on how probabilistic tests such as NPM (Numerical Probabilistic Method) can be used as an efficient alternative for GCS and decomposition when they are used with a good initial configuration (which we refer to as the witness configuration in Section 3.9). Section 4 considers GCS when there is a continuum of solutions and the use of curve tracing algorithms. Section 5 presents the expression of geometric constraints in a coordinate free way and shows how kernel functions can be used to provide intrinsic formulation of constraints. Section 6 discusses the need for formal specifications of constraints and for specification languages. The conclusion is given in Section 7.

2 Interval arithmetic and Bernstein bases

A recurrent problem of interval methods is the wrapping effect [Neumaier Cambridge, 2001]: interval arithmetic loses the dependance between variables, so that the width of intervals increases with the computation depth. Maybe Bernstein bases can help. They are well known in the CAD/CAM world, since Bézier and de Casteljau, but people in the interval analysis seem unaware of them. Fig. 1 permits to compare the naive interval arithmetic with the tensorial Bernstein based one: the same algebraic curve $f(x,y) = 0$ is displayed, with the same classical recursive method, using (above) the naive interval arithmetic and (below) the Bernstein interval arithmetic: clearly, the former needs much more subdivisions than the latter. Transcendental functions are a difficulty, of course. Either we enclose transcendental functions between some polynomials, using for instance a Bernstein-Taylor form as Nataraj and Kotecha [Nataraj and Kotecha 2004], or maybe the Poisson base is a solution [Morin 2001].

*e-mail: {dmichel,sfoufou,loic.lamarque}@u-bourgogne.fr
†e-mail:schreck@dpt-info.u-strasbg.fr

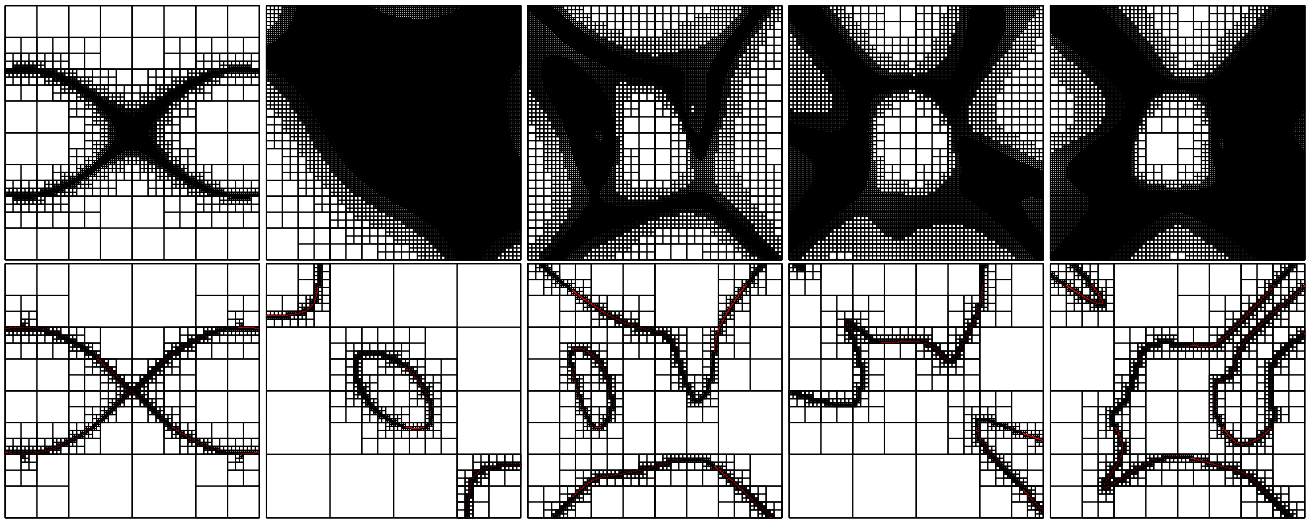

Figure 1: *Above*: naive interval arithmetic. *Below*: Bernstein based arithmetic. *Left to right columns*: Cassini oval: $C_{2,2}(x,y) = 0$ in $[-2,2] \times [-2,2]$, where $C_{a,b}(x,y) = ((x+a)^2+y^2) \times ((x-a)^2+y^2) - b^4$. The curve $f(x,y) = 15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ on the square $[0,1] \times [0,1]$. Random algebraic curves with total degree 10, 14, 18.

2.1 Tensorial Bernstein base

This section gives a flavor of tensorial Bernstein bases on a simple example of a polynomial equation $f(x,y) = 0,\ 0 \leq x,y \leq 1$. We consider $f(x,y) = 0$ as the intersection curve between the plane $z = 0$ and the surface $z = f(x,y)$. Assume f has degree 3 in x and y. Usually the polynomial f is expressed in the canonical base: $(1,x,x^2,x^3) \times (1,y,y^2,y^3)$, but we prefer the tensorial Bernstein base: $(B_{0,3}(x),B_{1,3}(x),B_{2,3}(x),B_{3,3}(x)) \times (B_{0,3}(y),B_{1,3}(y),B_{2,3}(y),B_{3,3}(y))$. The conversion between the two bases is a linear transform:

$$(B_{0,3}(x),B_{1,3}(x),B_{2,3}(x),B_{3,3}(x)) = (1,x,x^2,x^3) \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \quad (1)$$

and idem for y. This kind of formula and matrix extends to any degree: for degree n, the Bernstein base $B(t) = (B_{0,n}(t),B_{1,n}(t),\ldots B_{n,n}(t))$ and the canonical base $T = (1,t,t^2 \ldots t^n)$ are related by:

$$\begin{pmatrix} n \\ i \end{pmatrix} t^i = \sum_{j=i}^{n} \begin{pmatrix} j \\ i \end{pmatrix} B_{j,n}(t)$$

and

$$B_{i,n}(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i} = \sum_{j=i}^{n} (-1)^{j-i} \begin{pmatrix} n \\ j \end{pmatrix} \begin{pmatrix} j \\ i \end{pmatrix} t^j$$

The surface $z = f(x,y), 0 \leq x,y \leq 1$ has this representation in the Bernstein base:

$$x = \sum_{i=0}^{i=n} \frac{i}{n} B_{i,n}(x);\ y = \sum_{j=0}^{j=m} \frac{j}{m} B_{j,m}(y);\ z = \sum_{i=0}^{i=n} \sum_{j=0}^{j=m} z_{i,j} B_{i,n}(x) B_{j,m}(y)$$

Points $P_{i,j} = (\frac{i}{n}, \frac{j}{m}, z_{i,j})$ are called control points of the surface $z = f(x,y)$, which is now a Bézier surface. Control points have an intuitive meaning: typically, geometric modelers of curves and surfaces permit the user to interactively move control points and the Bézier curve or surface follows. A crucial property is that the surface patch (the part for $0 \leq x,y \leq 1$) lie inside the convex hull of its control points. Of course the convex hull of $f(0 \leq x \leq 1, 0 \leq y \leq 1)$ is just the interval $[\min z_{i,j}, \max z_{i,j}]$. It gives an enclosing interval for $f(0 \leq x \leq 1, 0 \leq y \leq 1)$, which is often sharper than the intervals provided by other interval arithmetics.

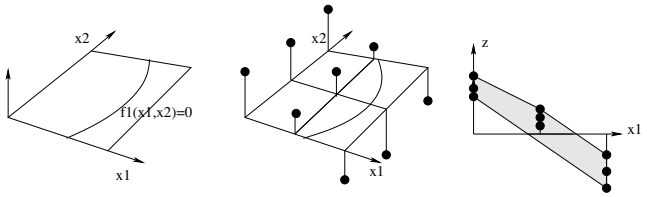

Figure 2: Equation $f_1(x_1,x_2) = 0$ has degree 2 in x and y, and a grid of 3×3 control points. The surface lie inside the convex hull of its control points. Computing the smallest rectangle enclosing the intersection between the plane $z = 0$ and this convex hull is a linear programming problem.

The method to display algebraic curves follows: if $0 \notin [\min_{i,j} z_{i,j}, \max_{i,j} z_{i,j}]$ then the curve does not cut the unit square, otherwise subdivide the square in four; the recursion is stopped at some recursion threshold; the Casteljau subdivision method permits to quickly compute the Bernstein representation (*i.e.* the control points) of the surface for any x interval $[x_0,x_1]$ and any y interval $[y_0,y_1]$, without translation to the canonical base. All that extends to higher dimension and the solving of systems of polynomial equations [Garloff and Smith 2001b; Garloff and Smith 2001a; Mourrain et al. 2004].

To find the smallest x interval $[x^-,x^+]$ enclosing the curve $f(x,y) = 0, 0 \leq x,y \leq 1$, project all control points on the plane x,z; compute their convex hull (it is an easy 2D problem); compute its intersection with the x axis: it is $[x^-,x^+]$. This is visually obvious on Fig. 2. Proceed similarly to find the smallest y-interval. In any dimension d, reducing the box needs only d computations of 2D con-

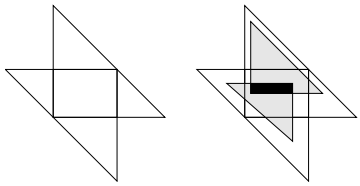

Figure 3: Reduction of a 2D box: it is the intersection of two triangles; reduce in the two triangles and take the bounding box.

vex hulls. A variant replaces the 2D convex hull computation by the computation of the smallest and greatest roots of two univariate polynomials, a lowest one and a largest one.

This box reduction is very advantageous when solving [Mourrain et al. 2004; Hu et al. 1996; Sherbrooke and Patrikalakis 1993] an algebraic system $f(x,y) = g(x,y) = 0, 0 \leq x,y \leq 1$ (or a more complex one), since it reduces the search space without subdivision or branching. Box reduction is even more efficient when combined to preconditionning: the system $f(x,y) = g(x,y) = 0$ has the same roots as a linear combination $af(x,y) + bg(x,y) = cf(x,y) + dg(x,y) = 0$; the idea is to use a linear combination such that $af(x,y) + bg(x,y)$ is very close to x and $cf(x,y) + dg(x,y)$ is very close to y: this combination is given by the jacobian inverse at the center of the considered box. It straightforwardly extends to higher dimension. Near a regular root, the convergence of such a solver is quadratic.

2.2 Simplicial Bernstein base

However there is a difficulty in high dimension: the tensorial Bernstein base has an exponential number of coordinates (as the canonical base) and is dense, *i.e.* a polynomial which is sparse in the canonical base becomes dense in the tensorial Bernstein base. For instance, a linear polynomial in d variables is represented by 2^d control points, a polynomial with total degree 2 is represented by 3^d control points, a polynomial with total degree n is represented by $(n+1)^d$ control points. A solution is to use the *simplicial* Bernstein base [Farin 1988] (the previous Bernstein base is the *tensorial* one).

For three variables x,y,z related by $x+y+z = 1$, the simplicial Bernstein base is defined by:

$$(x+y+z)^n = 1^n = \sum_{i+j+k=n} \binom{n}{i,j,k} x^i y^j z^k = \sum_{i+j+k=n} b^{(n)}_{ijk}(x,y,z)$$

and for any number of variables $x_1, \ldots x_d$ related by $x_1 + \ldots x_d = 1$, it is defined by:

$$(x_1 + x_2 + \ldots x_d)^n = 1^n = \sum_{i_1+\ldots i_d=n} \binom{n}{i_1,\ldots i_d} x_1^{i_1} \ldots x_d^{i_d} = \sum_{i_1+\ldots i_d=n} b^{(n)}_{i_1\ldots i_d}(x_1,\ldots x_d) \quad (2)$$

thus it straightforwardly extends the tensorial base defined by:

$$(x_k + (1-x_k))^n = 1^n = \sum_{i=0}^{i=n} \binom{n}{i} x_k^i (1-x_k)^{n-i} = \sum_{i=0}^{i=n} B_{i,n}(x_k), \quad k = 1,\ldots d$$

In the simplicial Bernstein base, a multivariate polynomial in d variable and with total degree n is represented by $O(d^n)$ control points; thus with total degree 1, 2, 3, etc, there are $O(d)$, $O(d^2)$, $O(d^3)$, etc control points. If the initial system is sparse in the canonical base, adding a logarithmic number of auxiliary unknowns and equations (using iterated squaring), every equation of any total degree $n \geq 2$ is translated into equations with total degree 2: thus with the simplicial Bernstein base the number of control points is polynomial; moreover the good properties of the tensorial Bernstein base still hold with the simplicial one: the convex hull property, the possibility of preconditioning, the possibility of reduction (it only needs several 2D convex hull problems as well), the de Casteljau method. An open question, which seems tractable, is which edge of the simplex to bissect?

2.3 Box reduction

For the moment, nobody uses the simplicial Bernstein base to solve algebraic systems. Perhaps it is due to the fact that domains are no more boxes (vectors of intervals) but simplices, which are less convenient for the programmer. In this respect, the simplicial Bernstein base can be used in a temporarily way, to reduce usual boxes, as follows. See Fig. 3 for a 2D example. A box $B = [x_1^-, x_1^+], \ldots [x_d^-, x_d^+]$ is given, it is cut by an hypersurface $H : h(x_1, \ldots x_d) = 0$; the problem is to reduce the box B as much as possible, so that it still encloses $B \cap H$. In any dimension d, the box is the intersection of two d simplices. Consider the hypercube $[0,1]^d$ for simplicity: a first simplex has $(0,0\ldots0)$ as vertex, the opposite hyperplane is $x_1 + x_2 + \ldots x_d = d$, its other hyperplanes are $x_i = 0$. The second simplex has vertex $(1,1,\ldots1)$, the opposite hyperplane is $x_1 + x_2 + \ldots x_d = 0$, its other hyperplanes are $x_i = 1$. To reduce the box, reduce in the two simplices, and compute the bounding box of the intersection between the two reduced simplices.

3 Decomposition related problems

3.1 DoF counting

All graph-based methods to decompose a system of geometric constraints rely on some DoF counting.

It is simpler to explain first the principle of DoF counting for systems of algebraic equations. A system of n algebraic equations is structurally well constrained if it involves n unknowns, also called DoF, and no subset of $n' < n$ equations involves less than n' unknowns, *i.e.* it contains no over-constrained part. For example, the system $f(x,y,z) = g(z) = h(z) = 0$ is not well constrained, because the subsystem $g(z) = h(z) = 0$ over-constrains z; remark that the details of f,g,h do not matter. Second example: the system $f(x,y) = g(x,y) = 0$ is structurally well constrained, *i.e.* it has a finite number of roots for generic f and g; if the genericity assumption is not fulfilled, it can have no solution: $g = f+1$, or a continuum of solutions: $f = g$. Later we will see that a pitfall of graph based approaches is that the genericity condition is not fulfilled.

A natural bipartite graph is associated to every algebraic system $F(X) = 0$. The first set of vertices represent equations: one equation per vertex. The second set of vertices represent unknowns: one unknown per vertex. An edge links an equation-vertex and an unknown-vertex iff the unknown occurs in the equation. The structural well-constrainedness of a system is equivalent to the existence of a complete matching in the associated bipartite graph (König-Hall theorem): a matching is a set of edges, with at most

one incident edge per vertex; vertices with an edge in the matching are said to be covered or saturated by the matching; a matching is maximum when it is maximum in cardinality; it is perfect iff all vertices are saturated. In intuitive words, one can find one equation per unknown (the two vertices are linked in the bipartite graph) which determines this unknown. There are fast methods to compute matchings in bipartite graph. Maximum matchings are equivalent to maximum flows (a lot of papers about graph based decomposition refer to maximum flows rather than maximum matchings).

The decomposition of bipartite graphs due to Dulmage and Mendelsohn also relies on maximum matchings. It partitions the system into a well-constrained part, an over-constrained part, an under-constrained part. The well-constrained part can be further decomposed (in polynomial time also) into irreducible well-constrained parts, which are partially ordered: for example $f(x) = g(x,y) = 0$ is well constrained; it can be decomposed into $f(x) = 0$ which is well constrained, and $g(x,y) = 0$ which is well constrained once x has been replaced by the corresponding root.

Relatively to systems of equations, systems of geometric constraints introduce two complications:

First geometric constraints are (classically...) assumed to be independent of the coordinate system, thus they can, for instance, determine the shape of a triangle in 2D (specifying either two lengths and one angle, or one angle and two lengths, or three lengths) but they can not fix the location and orientation of the triangle relatively to the cartesian frame. This placement is defined by three parameters (an x translation, an y translation, one angle). Thus in 2D, the DoF of a system is the number of unknowns (coordinates, radii, non geometric unknowns) minus 3. The same holds in 3D, where the placement needs six parameters; thus the DoF of a 3D system is the number of unknowns minus 6 — the constant is $d(d+1)/2$ in dimension d. Numerous ways have been proposed for adapting decomposition methods for systems of equations to systems of geometric constraints.

Second, the bipartite graph is visually cumbersome and not intuitive. People prefer the "natural" graph: each vertex represent a geometric unknown (point, line, plane) or a non geometric unknown, and each edge represents a constraint. There is a difficulty for representing constraints involving more than 2 entities; either hyper-arcs are used, or all constraints are binarized. Moreover vertices carry DoF, and edges (constraints) carry DoR: degree of restriction, *i.e.* the number of corresponding equations.

The differences between the bipartite and natural graphs are not essential. In passing, the matroid theory provides yet another formalism to express the same things, but it is not used in the GCS community up to now.

In 2D, a point and a line have 2 DoF; in 3D, points and planes have 3 DoF, lines have 4. In 3D, DoF counting (correctly) predicts there is a finite number of lines which cut four given skew lines: the unknown line has 4 DoF and there are 4 constraints. Similarly, there is a finite number of lines tangent to 4 given spheres.

Decomposition methods are essential, since they permit to solve big systems of geometric constraints, which can not be otherwise.

3.2 Decomposition modulo scaling or homography

Decomposition methods are complex and do not always take into account non geometric unknowns or geometric unknowns such as radii (which are independent on the cartesian frame, contrarily to unknowns). Decomposition methods should be simpler, more general, and decompose not only in subparts well constrained modulo displacements, but also modulo scaling [Schramm and Schreck 2003]: so we can compute an angle, or a distance ratio, in one part, and propagate this information elsewhere, and modulo homography: so we can compute cross ratios in one part and propagate elsewhere.

3.3 Almost decomposition

Hoffmann, Gao and Yang [Gao et al. 2004] introduce almost decompositions. They remark that a lot of irreducible systems in 3D are easier to solve when one of the unknowns is considered as a parameter, and when the system is solved for all (or some sampling) values of this parameter. In this artificial but simple example:

$$\begin{aligned}
f_1(x_1,x_2,x_3,x_4 = u) &= 0\\
f_2(x_1,x_2,x_3,x_4 = u) &= 0\\
f_3(x_1,x_2,x_3,x_4 = u) &= 0\\
f_4(x_1,x_2,x_3,x_4,x_5,x_6,x_7) &= 0\\
f_5(x_1,x_2,x_3,x_4,x_5,x_6,x_7) &= 0\\
f_6(x_1,x_2,x_3,x_4,x_5,x_6,x_7) &= 0\\
f_7(x_1,x_2,x_3,x_4,x_5,x_6,x_7) &= 0
\end{aligned}$$

x_4 is considered as a parameter u with a given value (we call it a key unknown for convenience). The subsystem $S_u : f_1(x_1,x_2,x_3,u) = f_2(x_1,x_2,x_3,u) = f_3(x_1,x_2,x_3,u) = 0$ is solved for all values of u, or in a more realistic way for some sampling of u. Then the rest of the system $T_u : f_4(x) = f_5(x) = f_6(x) = 0$ is solved, forgetting temporarily one equation, say f_7. f_7 is then evaluated at all sampling points on the solution curve of $f_1(x) = \ldots f_6(x) = 0$. When f_7 almost vanishes, the possible root is polished with some Newton iterations. For the class of basic 3D configurations studied by Hoffmann, Gao and Yang [Gao et al. 2004], one key unknown is sufficient most of the time, but some rare more difficult problems need two key unknowns. One may imagine several variants of this approach, for instance the use of marching curve methods to follow the curve parameterized with u, or methods to automatically produce the best almost decomposition for irreducible systems: the best is the one which minimizes the number of key unknowns.

Curve tracing [Michelucci and Faudot 2005] can also be used to explore a finite set of solutions when no geometric symbolic solution is available (which is often the case in 3D). If the solution proposed by the solver does not fit the user needs, the idea is to forget one constraint and to trace the corresponding curve. In this case the roots are the vertices of a graph the edges of which correspond to the curves where a constraint has been forgotten. If we have d equations and d unknowns then each vertex is of degree d. One difficulty is that this graph can be disconnected, and there is no guarantee to reach every vertex starting from a given solution.

3.4 Some challenging problems

Some challenging problems resist this last attack of almost decomposition. Consider the graph of the regular icosahedron (20 triangles, 30 edges, 12 vertices). Labelling edges with lengths gives a well constrained system with 30 distance constraints between 12 points in 3D (the regular pentagons of the icosahedron are not constrained to stay coplanar). This kind of problems, with distance constraints only, is called the molecule [Hendrickson 1992; Porta et al. 2003; Laurent 2001] problem because of its applications in chemistry: find the configuration of a molecule given some distances between its atoms. This last system has Bézout number $2^{30} \approx 10^9$.

A seemingly more difficult problem uses the graph of the regular dodecahedron (12 pentagonal faces, 20 vertices, 30 edges). Label edges with lengths; this time, also impose to each of the 12 pentagonal faces to stay planar, for the problem to be well constrained. The dodecahedron problem is not a molecule one, because of the coplanarity constraints. In the same family, the familiar cube gives a well constrained problem, with 8 unknown points, 12 distances, and 6 coplanarity relations.

Considering the regular octahedron gives a simpler molecule problem, with 6 unknown points and 12 distances (no coplanarity condition). This problem was already solved by Durand and Hoffmann [Durand 1998; Durand and Hoffmann 2000] with homotopy. Another solution is to use Cayley-Menger relations [Yang 2002; Porta et al. 2003; Michelucci and Foufou 2004].

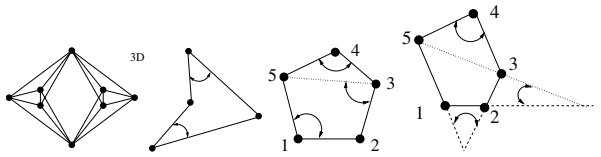

Figure 4: The double banana, and three other 3D configurations due to Auxkin Ortuzar, where DoF counting fails. No four points are coplanar.

3.5 Pitfalls of graph based methods

A pitfall of DoF counting is that geometric constraints can be dependent in subtle ways. In 2D, the simplest counter example to DoF counting is given by the 3 angles of a triangle: they can not be fixed independently (note they can in spherical geometry). Fig. 5 shows a more complex 2D counter example. In 3D, a simple counter example is: point A and B lie on line L, line L lie on plane H, point A lie on plane H; the last constraint is implied by the others. Fig. 4 shows other counter examples which make fail DoF counting in 3D. It is possible to use some ad hoc tests in graph based methods to account for some of these configurations. However every incidence theorem (Desargues, Pappus, Pascal, Beltrami, Cox... see Fig. 6) provide dependent constraints: just use its hypothesis and conclusion (or its negation) as constraints; moreover no genericity assumption (used in Rigidity theory) is violated since incidence constraints do not use parameters. Thus detecting a dependence is as hard as detecting or proving geometric theorems.

DoF counting is mathematically sound only in a very restricted case, the 2D molecule problem, *i.e.* when all constraints are generic distance constraints between 2D points (thus points can not be collinear): it is Laman theorem [J. Graver 1993]. For the 3D molecule problem, no characterization is known; Fig. 4 leftmost shows the most famous counterexample to DoF counting: the double banana, which uses only distance constraints. Even in the simple case of distance constraints, a combinatorial (*i.e.* in terms of graph or matroids) characterization of well constrainedness seems out of reach.

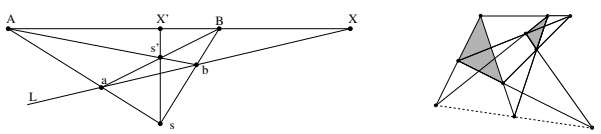

Figure 5: *Left*: be given 3 aligned points A,B,X; for any point s outside AB, for any L through X outside s, define: $a = L \cap As$, $b = L \cap Bs$, $s' = Ab \cap aB$, $X' = ss' \cap AB$; then X' is independent of s and L. *Right*: Desargues theorem: if two triangles (in gray) are perspective, homologous sides cut in three collinear points.

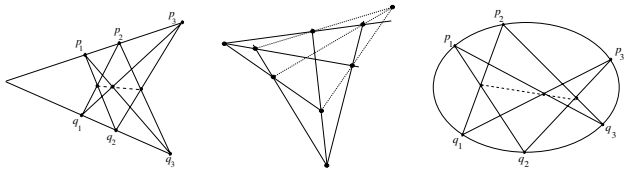

Figure 6: Pappus, its dual, Pascal theorems.

With the still increasing size of constraints systems, the probability for a subtle dependence increases as well. J-C. Léon (personal communication), who uses geometric constraints to define constrained surfaces or curves, reports this typical behavior: the solver detects no over-constrainedness but fails to find a solution; the failure persists when the user tries to modify the value of parameters (distances, angles) – which is terribly frustrating. This independence to parameter values suggests that the dependence is due to some incidence theorems of projective geometry (such as Pappus, Desargues, Pascal, Beltrami, etc). For conciseness, the other hypothesis: some triangular (or tetrahedral [Serré 2000]) inequality is violated, is not detailed here.

Detecting such dependences -or solving in spite of them when it is a consistent dependence- is a key issue for GCS. Clearly, no graph based method can detect all such dependences. It gives strong motivation for investigating other methods.

3.6 Linear Algebra performs qualitative study

Today decomposition is graph based most of the time. Linear algebra seems a promising alternative. For conciseness, the idea is illustrated for 2D systems of distance constraints only between n points. Assume also the distances are algebraically independent (thus no collinear points), and that points are represented by their cartesian coordinates: $X = (x_1, y_1, \ldots x_n, y_n)$. For clarity, we say that $p = (x, y)$ is a "point", and X is a "configuration". After Rigidity theory [J. Graver 1993; Lamure and Michelucci 1998], it is well known that it suffices to numerically study the jacobian at some random configuration $X \in \mathbb{R}^{2n}$. It is the essence of the so called numerical probabilistic method (NPM).

By convention, the k th line of the jacobian J is the derivative of the k th equation of the system. Vectors m such that $Jm^t = 0$ are called infinitesimal motions. The notation $\dot{X} = (\dot{x_1}, \dot{y_1}, \ldots \dot{x_n}, \dot{y_n})$ is also used to denote the infinitesimal motion at configuration X.

First, if the rank of the jacobian (at the random, generic configuration) is equal to the number of equations, equations are independent; otherwise it is possible to extract a base of the equations. Second, the system is well-constrained (modulo displacement) if its jacobian has corank 3: actually it is even possible to give a base of the kernel of the jacobian (the kernel is the set of infinitesimal motions). This base is t_x, t_y, r, where $t_x = (1,0,1,0,\ldots)$ is a translation in x, $t_y = (0,1,0,1,\ldots)$ is a translation in y, and r is an instantaneous rotation around the origine: $r = (-y_1, x_1, -y_2, x_2, \ldots - y_n, x_n)$. These 3 infinitesimal motions are displacements, also called isometries; they do not modify the relative location of points, contrarily to deformations (also called flexions).

An infinitesimal motion m is a displacement iff for all couple of points A,B, the difference $\dot{A} - \dot{B}$ between A and B motions is or-

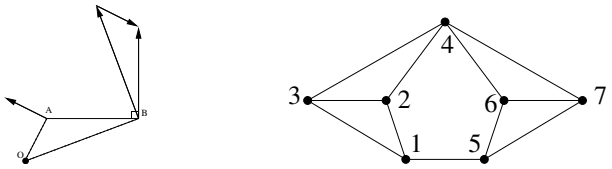

Figure 7: *Left*: the arrows illustrate the infinitesimal rotation around O of points A and B. For a displacement like this rotation, $\dot{A}-\dot{B}$ is orthogonal to AB for all couples A,B. *Right*: this system is well-constrained. Removing the bar (the constraint distance) $1,5$ breaks the system into two well-constrained parts (the left and the right of point 4).

thogonal to the vector $\overrightarrow{AB}$. Fig. 7 illustrates that for the rotation r.

For convenience, define $d_{i,j}$ as the vector $(\dot{x_1},\dot{y_1},\ldots\dot{x_n},\dot{y_n})$ where $\dot{x_i}=x_i-x_j$, $\dot{x_j}=x_j-x_i$, $\dot{y_i}=y_i-y_j$, $\dot{y_j}=y_j-y_i$ and $\dot{x_k}=\dot{y_k}=0$ for $k\neq i,j$; actually, $d_{i,j}$ is half the derivative of the distance equation $(x_i-x_j)^2+(y_i-y_j)^2-D_{ij}^2=0$. Obviously $d_{i,j}=-d_{j,i}$. It is easy to check that, consistently, t_x, t_y and r are orthogonal to all $d_{i,j}, 1\leq i<j\leq n$: they indeed are displacements, not deformations.

All that is well known, after Rigidity theory [J. Graver 1993]. What seems less known is that linear algebra makes also possible to decompose a well-constrained system into well-constrained subparts.

3.7 The NPM decomposes

Consider for instance the well-constrained system in Fig. 7 Right, and remove the constraint distance (the edge) $1,5$. It increases the corank by 1, adding an infinitesimal flexion (a deformation); a possible base for the kernel is t_x,t_y,r and $f=(0,0,0,0,0,0,0,0,y_4-y_5,x_5-x_4,y_4-y_6,x_6-x_4,y_4-y_7,x_7-x_4)$ *i.e.* an instantaneous rotation of $4,5,6,7$ around 4, or $g=(y_4-y_1,x_1-x_4,y_4-y_2,x_2-x_4,y_4-y_3,x_3-x_4,0,0,0,0,0,0,0,0)$ *i.e.* an instantaneous rotation of $1,2,3,4$ around 4, or any linear combination m of f, g, t_x,t_y,r (outside the range of t_x,t_y,r, to be pedantic). Of course f and g especially make sense for us, but any deformation m is suitable.

The deletion of edge $1,5$ leaves the part 1,2,3,4 well-constrained: it is visually obvious, and confirmed by the fact that $d_{i,j}, 1\leq i<j\leq 4$ is orthogonal to m. Idem for the part 4,5,6,7, because $d_{i,j}, 4\leq i<j\leq 7$ is orthogonal to m. But no $d_{i,j}$ with $i<4<j$ is orthogonal to m. This gives a polynomial time procedure to find maximal (for inclusion) well-constrained parts in a flexible system, and a polynomial time procedure to decompose well-constrained systems into well-constrained subsystems: remove a constraint and find remaining maximal (for inclusion) well-constrained parts, as in the previous example.

This idea can be easily extended to 3D distance constraints, with some minor changes: the corank is 6 instead of 3. Note this method detects the bad constrainedness of the classical double banana, contrarily to graph based methods which extend the Laman condition.

What if other kinds of constraints are used, not only distance constraints? From a combinatorial point of view, the vertices in Fig. 7 can represent points, but also lines (which have also 2 DoFs, like points, in 2D). Thus as far as decomposing an well-constrained graph into well-constrained subparts is concerned, we can consider vertices of the graph as points, and constraints/edges as distance constraints. This first answer is not always satisfactory, for instance when vertices have distinct DoF (in 3D, points and planes have 3 DoF, but lines have 4), or when constraints involve more than 2 geometric objects.

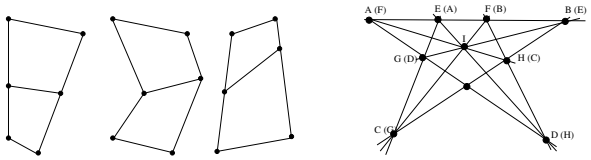

Figure 8: *From left to right*: the unknown solution configuration; a random configuration, not fulfiling incidence constraints; a witness configuration; an irrational configuration with an underlying regular pentagon (or an homography of).

In fact this method has been extended to other kind of constraints [Foufou et al. 2005]. The only serious difficulty occurs when the assumption of the genericity of the relative location of points is contradicted by some explicit (or induced) projective constraints (collinearity or coplanarity constraints). Of course graph based decomposition methods have the same limitation.

3.8 The witness configuration principle

Clearly, the NMP give incorrect results because it studies the jacobian at a random, generic, configuration which does not fulfil these projective constraints. A solution straightforwardly follows: compute a "witness configuration" and study it with the NPM; a witness configuration [Foufou et al. 2005; Michelucci and Foufou 2006] does not satisfy the metric constraints (*i.e.* it has typically lengths or angles different of the searched configuration), but it fulfils the specified projective constraints (see Fig. 9), and also, by construction, the projective constraints (collinearities, coplanarities) due to geometric theorems of projective geometry, *e.g.* Pascal, Pappus, Desargues theorems. First experiments validate the witness configuration method [Foufou et al. 2005]: it works for all counter examples to DoF in this paper (for instance Fig. 4 or 5), and it is even able to detect and stochastically prove incidence theorems which confuse DoF counting (see below). In other words no confusing witness configuration has been found up to now.

3.9 Computing a witness configuration

Most of the time, the sketch is a witness configuration. Otherwise, if the distance and angle parameters are generic (no right angle, for instance), remove all metric constraints and solve the remaining (very under-constrained system); the latter contains only projective constraints, *i.e.* incidence constraints. Even for the challenging problems: icosahedron, dodecahedron, cube, it is trivial to find a witness polyhedron – the latter can be concave or self intersecting.

Finally, if distance and angle values are not generic (*e.g.* right angles are used), the simplest strategy is to consider parameters as unknowns (systems are most of the time of the form: $F(U,X)=0$ where U is a vector of parameters: lengths, angle cosines, etc; their values are known just before resolution), then to solve the very under-constrained resulting system: it is hoped it is easily solvable. Once a solution has been found, it gives a witness configuration which is studied and decomposed with the NPM.

This section has given strong motivations to study the decomposition and resolution of under-constrained systems, and of systems of incidence constraints.

3.10 Incidence constraints

The previous section has already given some motivations to study incidence constraints, but these constraints also arise in photogrammetry, in computer vision, in automatic correction of hand made drawings. We hope the systems of incidence constraints met in our applications to be trivial or almost trivial (defined below), however incidence constraints can be arbitrarily difficult even in 2D.

In 2D, a system of incidence constraints between points and lines reduce to a special 3D molecule problem [Hendrickson 1992; Porta et al. 2003; Laurent 2001]: represent unknown points and lines by unit 3D vectors; the incidence $p \in L$ means that the corresponding vertices on the unit sphere have distance $\sqrt{2}$. To avoid degeneracies (either all points are equal, or all lines are equal), one can impose to four generic points to lie on some arbitrary square on the unit sphere.

3.10.1 Trivial and almost trivial incidence systems

In 2D, a system of incidence constraints (point-line incidences) is trivial iff it contains only removable points and lines. A point p is removable when it is constrained to lie on two lines l_1 and l_2 (or less): then its definition is stored in some data structure (either $p = l_1 \cap l_2$, or $p \in l_1$ is any point on line l_1, or p is any point), it is erased from the incidence system, the rest of the system is solved, then the removed point is added using its definition. Symmetrically (or dually) for a line, when it is constrained to pass through two points (or less). Erasing a point or a line may make removable another point or line. If all points and lines are removed, the graph is trivial. Trivial systems are easily solved, using the definitions of removed elements in reverse order.

The extension to 3D is straightforward. This method finds a witness for every Eulerian 3D polyhedra (a polyhedron is Eulerian iff it fulfils Euler formula). It is easily proved that every Eulerian 3D polyhedron contain a removable vertex or a removable face, and thus is trivial: assume there is a contradicting polyhedron, with V vertices, E edges and F faces. Let $v_1, v_2 \dots v_V$ be the vertex degrees, all greater than 3, and $f_1, f_2 \dots f_F$ the number of vertices of the F faces, all greater than 3 as well; it is well known that $\sum_1^V v_i = 2E = \sum_1^F f_j$, thus $E \geq 2V$ and $E \geq 2F$. By Euler' formula: $V - E + F = 2$. Thus $E + 2 = V + F \geq 2V + 2$ and $E + 2 = V + F \geq 2F + 2$. Add. We get $2E + 4 = 2V + 2F \geq 4 + 2V + 2F$: a contradiction. QED. Unfortunately, this simple method no more applies with non Eulerian polyhedra, say a faceted torus with quadrilateral faces and where every vertex has degree 4 (this last polyhedron has a rational witness too).

Another construction of a witness for Eulerian polyhedra first computes a 2D barycentric embedding (also called a Tutte embedding) of its vertices and edges: an arbitrary face is mapped to a convex polygon and other vertices are barycenters of their neighbors – it suffices to solve a linear system. Maxwell and Cremona already knew that such a 2D embedding is the projection of a 3D convex polyhedron; for instance, the three pairwise intersection edges of the three faces of a truncated tetrahedron concur. It is then easy to lift the Tutte embedding to a 3D convex polyhedron, using propagation and continuity between contiguous faces. In passing, this construction proves Steinitz theorem: all 3D convex polyhedra are realizable with rational coordinates only, and thus with integer coordinates only; this property is wrong for 4D convex polyhedra [Richter-Gebert 1996].

Configurations in incidence theorems are typically almost trivial (the word is chosen by analogy with almost decomposition). A system is almost trivial iff, removing an incidence, the obtained system is trivial: Desargues, Pappus, hexamy[1] configurations are almost trivial.

Almost triviality permits the witness configuration method to detect and prove incidence theorems in a probabilistic way: erase an incidence constraint to make the system trivial; for Pappus, Desargues, hexamy configurations to quote a few, due to the symmetry of the system, every incidence is convenient; solve the trivial system.

- If the obtained configuration fulfils the erased incidence constraint, then this incidence is with high probability a consequence of the other incidences: a theorem has been detected and (probabilistically) proved. A prototype [Foufou et al. 2005], performing computations in a finite field $\mathbb{Z}/p\mathbb{Z}$ (p a prime, near 10^9) for speed and exactness, probabilistically proves this way all theorems cited so far and some others, such as the Beltrami theorem[2] in 3D in a fraction of a second. This shows that using some computer algebra is possible and relevant.
- If the obtained configuration does not fulfil the erased incidence constraint, this constraint is not a consequence of the others. This case occurs with the pentagonal configuration in Fig. 8; the later is not realizable in $\mathbb{Q}$: indeed a regular pentagon (or an homography of) is needed. This configuration is not relevant for CAD-CAM (actually, we know none).

3.10.2 Universality of point line incidences

However, incidence constraints in 2D (and a fortiori in 3D) can be arbitrarily difficult; it is due to the following theorem which is a restatement[3] of the fundamental theorem of projective geometry, known since von Staudt and Hilbert [Bonin 2002; Coxeter 1987; Hilbert 1971]:

Theorem 1 (Universality theorem) *All algebraic systems of equations with integer coefficients and unknowns in a field $\mathbb{K}$ (typically $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$) reduce to a system of point-line incidence constraints in the projective plane $\mathbb{P}(\mathbb{K})$, with the same bit size.*

The proof relies on the possibility to represent numbers by points on a special arbitrary line, and on the geometric construction (with ruler only) of the point representing the sum or the product of two numbers (Fig. 9), from their point representation [Bonin 2002]. Some consequences are:

- Alone, point-line incidences in the projective plane are sufficient to express all geometric constraints of today GCS.
- Programs solving point line incidence constraints (*e.g.* solving the 3D molecule problem [Hendrickson 1992; Laurent 2001; Porta et al. 2003]) can solve all systems of geometric constraints.
- Programs detecting or proving incidence theorems in 2D (as the hexamy prover [Michelucci and Schreck 2004]) address all algebraic systems. Fascinating.

[1] An hexamy is an hexagon the opposite sides of which cut in three collinear points; every permutation of an hexamy is also an hexamy; it is a desguise of Pascal theorem.

[2] Coxeter [Coxeter 1999; Coxeter 1987] credits Gallucci for this theorem, in his books.

[3] D. Michelucci and P. Schreck. Incidence constraints: a combinatorial approach. Submitted to the special issue of IJCGA on Geometric Constraints.

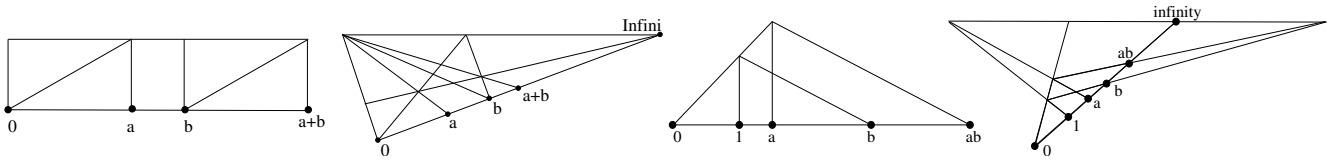

Figure 9: *left*: affine and projective constructions of $a+b$; *right*: affine and projective constructions of $a \times b$

- Algebra reduces to combinatorics: the bipartite graph of the point line incidences contains *all* the information of the algebraic system: no need for edge weights, no genericity assumption (contrarily to Rigidity theory).
- This bipartite graph is a fundamental data structure. What are its properties? its forbidden minors? Which link between its graph properties and properties of the algebraic system?
- Incidence constraints are definitively not a toy problem.

Practical consequences are unclear for the moment: for instance, does it make sense to reduce algebraic systems to a (highly degenerate) 3D molecule problem? Probably not.

4 Solving with a continuum of solutions

Current solvers assume that the system to be solved has a finite number of solutions, and get into troubles or fail when there is a continuum of solutions.

Arguably, computer algebra [Chou 1988; Chou et al. 1987], and geometric solvers (typically ruler and compass) already deal with under constrainedness; both are able to triangularize in some way given under-constrained systems $F(X) = 0$; for instance, several elimination methods from computer algebra are (at least theoretically) able to partition the set X of unknowns into $T \cup Y$, where T is a set of parameters, and to compute a triangularized system of equations: $g_1(T,y_1) = g_2(T,y_1,y_2) = \ldots g_n(T,y_1,y_2,\ldots y_n) = 0$ which define the unknowns in Y. In the 2D case, and when a ruler and compass construction is possible, some geometric solvers are able to produce a construction program (also named: straight line program, DAG, etc): $y_1 = h_1(T), y_2 = h_2(T,y_1), \ldots y_n = h_n(T,y_1 \ldots y_{n-1})$, where h_i are multi-valued functions for computing the intersection between two cercles, or between a line and a cercle, etc; dynamic geometry softwares [Bellemain 1992; Kortenkamp 1999; Dufourd et al. 1997; Dufourd et al. 1998] have popularized this last approach, which unfortunately does not scale well in 3D.

Another approach, typically graph-based, considers that under-constrainedness are due to a mistake from the user or to an incomplete specification; they try to detect and correct these mistakes, or to complete the system to make it well-constrained – and as simple to solve as possible [Joan-Arinyo et al. 2003; Gao and Zhang 2003; Zhang and Gao 2006].

Some systems are intrinsically under-constrained: the specified set is continuous. This happens when designing mechanisms or articulated bodies, when designing constrained curves or surfaces (for instance for blends), when using an almost decomposition, when searching a witness. Thus it makes sense to design more robust solvers, able to deal with a continuum of solutions. Such a solver should detect on the fly that there is a continuum of solutions, should compute the dimension of the solution set (0 for a finite solution set, 1 for a curve, 2 for a surface, etc) and should be able to segment solution curves and to triangulate solution surfaces, etc.

Methods for computing the dimension of a solution set already exist in computer graphics (and elsewhere); roughly, cover the solution set with a set of boxes (as in Fig. 1) with size length ε; if halving ε multiplies the number of boxes by about 1, 2, 4, 8, etc, induce that the solution set has dimension 0, 1, 2, 3, etc; this is the Bouligand dimension of fractals [Mandelbrot 1982; Barnsley 1998]. Instead of boxes for the cover, it is possible to use balls or simplices. This ultimate solver will unify the treatment of parameterized surfaces, implicit surfaces, blends, medial axis, and geometric constraints in geometric modeling. C. Hoffmann calls that the "dimensionality paradigm".

Fig. 10 illustrate such an ultimate solver with examples, mainly 2D for clarity. For the first picture, the input is the system:

$$\begin{cases} (x-x_c)^2+(y-y_c)^2 = r^2 \\ (x_1-x_c)^2+(y_1-y_c)^2 = r^2 \\ (x_2-x_c)^2+(y_2-y_c)^2 = r^2 \\ (x_3-x_c)^2+(y_3-y_c)^2 = r^2 \end{cases}$$

with x_n, y_n the coordinates of the triangle vertices and x, y, x_c, y_c, r the unknowns.

The second picture represents two circles with the radii defined by an equation; the input of the solver is the system:

$$\begin{cases} x^2+y^2 = r^2 \\ (r-1)(r-2) = 0 \end{cases}$$

The third one shows the section of a Klein's bottle; the input of the solver is:

$$\begin{cases} (x^2+y^2+z^2+2y-1)((x^2+y^2+z^2-2y-1)^2-8z^2)+ \\ \qquad 16xz(x^2+y^2+z^2-2y-1) = 0 \\ x-z = 1 \end{cases}$$

The latter is the intersection curve between an extruded folium and a sphere; the input of the solver is the system:

$$\begin{cases} x^2+y^2+z^2 = 1 \\ x^3+y^3-3xy = 0 \end{cases}$$

The two last pictures illustrate also an adaptive subdivision in accordance with the curvature of the solution set inside a box and a detection of the boxes containing singular points. In these examples, the Bouligand dimension is used also to get rid of terminal boxes (at the lowest subdivision depth) without solutions.

5 Coordinates-free constraints

Recently several teams [Yang 2002; Lesage et al. 2000; Serré et al. 1999; Serré et al. 2002; Serré et al. 2003; Michelucci and Foufou 2004] propose coordinate-free formulations, which are sometimes advantageous. For instance, the Cayley Menger determinant links

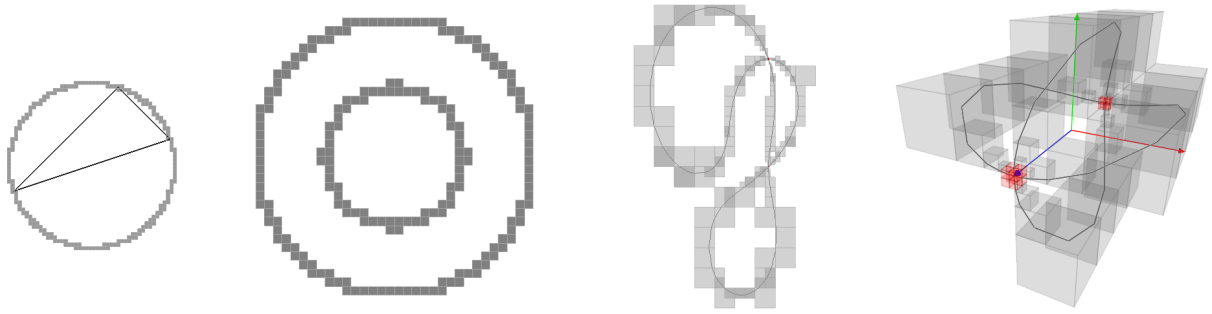

Figure 10: Some preliminary results of a solver based on centered interval arithmetic and Bouligand dimension; *left*: a triangle's circumscribed circle; *middle-left*: two circles with "unknown" radius; *middle-right*: intersection between a plan and the Klein's bottle; *right*: intersection between an extruded folium and a sphere.

the distances between $d+2$ points in dimension d and gives, for the octahedron problem, a very simple system solvable with Computer Algebra. These intrinsic relations have been extended to other configurations, *e.g.* with points and planes in 3D, points and lines in 2D. An intrinsic relation, due to Neil White, is given in Sturmfels's book [Sturmfels 1993], th. 3.4.7: it is the condition for five skew lines in $3D$ space to have a common transversal line. Philippe Serré, in his PhD thesis [Serré 2000], gives the relation involving distances between two lines AB and CD and between points A, B, C, D. However, for 3D configurations involving not only lines but also points or planes, intrinsic formulations (e.g. extending Cayley-Menger formulations) are missing most of the time. Even the intrinsic condition for a set of points to lie on some algebraic curve or surface with given degree was unknown (it is given just below). Next sections suggest methods to find such relations. These issues are foreseeable topics for GCS.

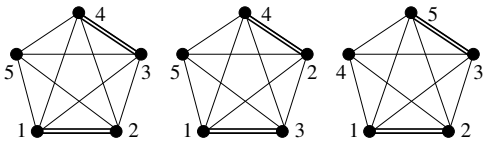

Figure 11: Isomorphic subgraphs of the same class monomials.

5.1 Finding new relations

To find relations linking invariants (distances, cosines, scalar products, signed areas or volumes *i.e.* determinants) for a given configuration of geometric elements, it suffices in theory to use a Grobner package which eliminates variables representing coordinates in some set of equations, for instance equations: $(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2-d_{ij}^2=0, i\in[1;4], j\in[i+1;5]$, to find the Cayley-Menger equation relating distances between 5 points in $3D$. In practice, computer algebra is not powerful enough. The polynomial conditions can be computed by interpolation: for instance, to guess the Cayley-Menger equation in $3D$, one can proceed in three steps: first, generate N random configurations of 5 points $(x_i,y_i,z_i)\in\mathbb{Z}^3$, second compute square distances $d_{ij}^{(k)}, i\in[1;4]$ and $j\in[i+1;5]$ for each configuration $k\in[1;N]$; this gives N points with 15 coordinates; third, all these N points lie on the zero-set of an unknown polynomial in the variables d_{ij}: search for this polynomial by trying increasing degrees.

This polynomial has an exponential number of monomials, thus an exponential number of unknown coefficients. Due to symmetry, some monomials have the same coefficients; they are said to lie in the same class. For instance, monomials $d_{12}^2d_{34}^2$, $d_{13}^2d_{24}^2$, etc lie in the same class; monomials in the same class correspond to isomorphic edge weighted subgraphs of K_5, the complete graph with 5 vertices and with edges weighted by the degree of the corresponding monomial (Fig. 11). To be feasible this interpolating method must exploit this symmetry. The fast generation of these classes (and of one instance per class) is an interesting and non trivial combinatorial problem by itself, related to the Reed-Polya counting theory. To validate this approach, we implemented a simple algorithm, which successfully computes Cayley-Menger relations, and distance relations for six 2D (ten 3D) points to lie on the same conic (quadric). A lesson of this prototype is that another good reason to exploit symmetry is to limit the size of the output interpolating polynomial.

5.2 Kernel functions provide intrinsic formulations

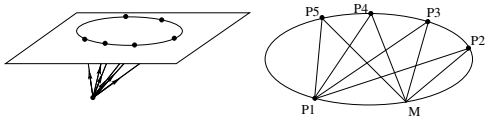

Figure 12: Vectorial condition for points to lie on a common conic or algebraic curve with degree d (two cases).

Given a set of non null vectors $v_i, i=1\ldots n$ having a common origin Ω, the set of lines l_i supported by these vectors and a plane π not passing through Ω (Fig. 12), what is the condition on the scalar products between vectors v_i for the intersection points between lines l_i and plane π to lie on the same conic? This section shows that the matrix M with $M_{i,j}=(v_i\cdot v_j)^2=M_{j,i}$ must have rank 5 or less. If the $n\geq 6$ intersection points do not lie on the same conic, but are generic, the matrix M has rank 6 (assuming the v_i lie in 3D space). More generally:

Theorem 2 *The intersection points between a plane π and the lines defined by supporting vectors v_i through a common origin Ω outside π lie on a degree d curve iff the matrix $M^{(d)}$, where $M_{i,j}^{(d)}=(v_i\cdot v_j)^d=M_{j,i}^{(d)}$, has rank $r_d=d(d+3)/2$ or less (r_d for deficient rank). The generic rank $g_d=r_d+1=(d+1)(d+2)/2$ (the rank of the matrix in the generic case) is given by the number of monomials in the polynomial in 2 variables of degree d, since this curve is the zero set of such a polynomial.*

The proof uses kernel functions [Cristianini and Shawe-Taylor 2000]. Let $p_i=(x_i,y_i,h_i), i=1\ldots 6$ be six homogeneous points

in 2D and ϕ_2 the function that maps each point p_i to $P_i = \phi_2(p_i) = (x_i^2, y_i^2, h_i^2, x_i y_i, x_i h_i, y_i h_i)$. By definition of conics, if points p_i lie on a common conic $ax_i^2 + by_i^2 + ch_i^2 + dx_i y_i + ex_i h_i + fy_i h_i = 0$, then points P_i lie on a common hyperplane, having equation: $P_i \cdot h = 0$ with $h = (a,b,c,d,e,f)$. Thus six generic (or random, *i.e.* not lying on a common conic) 2D points p_i give six lifted points P_i with rank 6, and six 2D points p_i lying on a common conic give six lifted points P_i with rank $r_2 = 5$ or less.

If m vectors $P_1, \ldots P_m$ have rank r, their Gram matrix $G_{ij} = P_i \cdot P_j = G_{ji}$ has also rank r. To compute $P_i \cdot P_j$, the naive method compute $P_i = \phi_2(p_i)$, and $P_j = \phi_2(p_j)$, then $P_i \cdot P_j$. Kernel functions avoid the computation of $\phi_2(p_i)$. A kernel function K is such that $K(p_i, p_j) = \phi_2(p_i) \cdot \phi_2(p_j)$.

A first example of kernel function considers given $p = (x,y,h)$ and homogeneous $\phi_2(p) = (x^2, y^2, h^2, \sqrt{2}xy, \sqrt{2}xh, \sqrt{2}yh)$. The cosmetic $\sqrt{2}$ constant does not modify rank but simplifies computations: $K(p,p') = \phi_2(p) \cdot \phi_2(p') = \phi_2(p) \cdot \phi_2(p') = \ldots = (p \cdot p')^2$ as the reader will check. More generally, for an homogeneous kernel polynomial of degree d, $K(p,p') = (p \cdot p')^d$: it suffices to adjust the cosmetic constants. Thus the Gram matrix for this homogeneous lift with degree d is: $G_{i,j} = (p_i \cdot p_j)^d$. The proof of the previous theorem follows straightforwardly.

A second example considers a non homogeneous lifting polynomial. Let $p = (x,y)$ and $\phi(p) = (x^2, y^2, \sqrt{2}xy, \sqrt{2}x, \sqrt{2}y, 1)$. As above, the $\sqrt{2}$ does not modify rank but simplifies computations: $K(p,p') = \phi(x,y) \cdot \phi(x',y') = \ldots = (p \cdot p' + 1)^2$ as the reader will check. More generally, for a non homogeneous lifting polynomial of degree d, $K_(p,p') = (p \cdot p' + 1)^d$. Thus the Gram matrix for this lift with degree d is $G_{i,j} = (p_i \cdot p_j + 1)^d$. We use the latter to answer the question: what is the coordinate-free condition for six 2D points $P_i, i = 0 \ldots 5$ to lie on a common quadric, or on a common algebraic curve with degree d? We search a condition involving scalar products between vectors P_0P_j, thus independent on the coordinates of points P_i. Suppose that the plane π containing points P_i is embedded in 3D space, let Ω be any one of the two points such that ΩP_0 is orthogonal to π, and the distance ΩP_0 equals 1. Use the previous theorem: points P_i lie on the same conic iff the matrix M, where $M_{i,j} = (\overrightarrow{\Omega P_i} \cdot \overrightarrow{\Omega P_j})^2$, has rank five or less, and the P_i lie on the same algebraic curve with degree d iff the matrix M, where $M_{i,j} = (\overrightarrow{\Omega P_i} \cdot \overrightarrow{\Omega P_j})^d$ has deficient rank $r_d = d(d+3)/2$. We remove Ω:

$$\begin{aligned}
\overrightarrow{\Omega P_i} \cdot \overrightarrow{\Omega P_j} &= (\overrightarrow{\Omega P_0} + \overrightarrow{P_0 P_i}) \cdot (\overrightarrow{\Omega P_0} + \overrightarrow{P_0 P_j}) \\
&= \overrightarrow{\Omega P_0} \cdot \overrightarrow{\Omega P_0} + \overrightarrow{\Omega P_0} \cdot \overrightarrow{P_0 P_j} + \overrightarrow{P_0 P_i} \cdot \overrightarrow{\Omega P_0} + \overrightarrow{P_0 P_i} \cdot \overrightarrow{P_0 P_j} \\
&= 1 + 0 + 0 + \overrightarrow{P_0 P_i} \cdot \overrightarrow{P_0 P_j}
\end{aligned}$$

Theorem 3 *Coplanar points P_i lie on the same algebraic curve with degree d iff the matrix M has deficient rank (*i.e. $r_d = d(d+3)/2$*) or less, where $M_{i,j} = (1 + \overrightarrow{P_0 P_i} \cdot \overrightarrow{P_0 P_j})^d$.*

These theorems nicely extends to surfaces and beyond. All relations involving scalar products can be translated into relations involving distances only, using: $\vec{u} \cdot \vec{v} = (\vec{u}^2 + \vec{v}^2 - (\vec{v} - \vec{u})^2)/2$.

6 The need for formal specification

Constraints systems are sets of specification described using some specification languages. Up to now, the community of constraint modeling focused more on solvers than on the study of description languages. However, the definition of such languages is of major importance since it sets up the interface between the solver, the modeler and the user.

On this account, a language of constraints corresponds to the external specifications of a solver: it makes the skeleton of the reference manual of a given solver, or, conversely, it defines the technical specifications for the solver to be realized. On the other hand, a language of constraints has to be clearly and fully described in order to be able to define the conversion from a proprietary architecture to an exchange format (which is itself described by such a language). CAD softwares are currently offering several exchange formats, but, in our sense, they are very poorly related to the domain of geometric constraints and they are unusable, for instance, for sharing benchmarks [Aut 2005].

It seems to us that a promising track in this domain consists in considering the *meta*-level. More clearly, we argue that we need a standard for the description of languages of geometric constraints rather than (or in addition to) specific exchange formats. This is the point of view adopted by the STEP consortium which is, as far as we know, not concerned by geometric constraints [4]. Besides, a meta-level approach allows to consider a *geometric universe* as a *parameter* of an extensible solver.

A first attempt in this direction was presented in [Wintz et al. 2005]. This work borrows the ideas and the terminology of the algebraic specification theory [Wirsing 1990; Goguen 1987]: a constraint system is syntactically defined by a triple (C,X,A) where X and A are some symbols respectively referring to unknowns and parameters, and C is a set of predicative terms built on a heterogeneous signature Σ. Recall that a heterogeneous signature is a triple $< S,F,P >$ where

- S is a set of *sorts* which are symbols referring to types,
- F is a set of *functional symbols* typed by their profile $f : s_1 \ldots s_m \rightarrow s$,
- P is a set of *predicative symbols* typed by their profile $p : s_1 \ldots s_k$

Functional symbols express the tools related to geometric construction while the predicative symbols are used to describe geometric constraints. The originality of the approach described in [Wintz et al. 2005] consists in the possibility of describing the semantic —or more precisely, several semantics like visualization, algebra, logic— within a single framework allowing to consider as many tool sets as provided semantic fields.

Since the main tools are based on syntactic analyzers, the support language considered is XML which is flexible enough to allow the description of geometric universes and which comes with a lot of facilities concerning the syntactic analysis.

The advantages of using formal specifications can be summarized in the two following points: (i) Clarifying and expliciting the semantics, which helps to avoid the misunderstandings that commonly occur between all the partners during data exchanges; and (ii) There are more and more software tools: parsers, but also provers, code generators, compilers, which are able to use these explicited semantics; these tools make it easier to ensure the reliability and the consistence between distinct pieces of software, to extend the software and to document it.

There is, of course, a lot of works to do in this domain, let us enumerate some crucial points:

[4] although some searchers are working on such a task (personal communication of Dr. Mike Pratt)

- the definition of tools able to describe and handle the translation between two languages of constraints (for instance using the notion of signature morphism).
- the automatic generation of tools from a given language of constraints.
- the possibility to take into account robustness consideration in the framework (see, for instance, [Schreck 2001])
- ideally, it is possible to imagine that such languages would be able to fully describe geometric solvers from the input of the constraints to the expression and visualization of the solutions

Tackling the problem from another point of view, the user, that is the designer, should be allowed to enter his proper solutions of a constraints system or his proper geometric constructions within the application. This should be made easier by considering a precise language of constraints. This family of tools come with the ability of compiling constructions and doing parametric design (see [Hoffmann and Joan-Arinyo 2002]). This problem is naturally very close to the generative modeling problem and the well known notion of features.

We think that the fields of geometric constraints solving and features modeling are mature enough for attempting to join them together (see for instance [Sitharam et al. 2006]). Indeed, the geometric constraints solving field addresses routinely 3D problems and takes more and more semantic aspects into consideration. This should give some hints to handle the problematic of under-constrained or over-constrained constraint systems. Indeed, up to now, researchers in GCS field have considered this problem from a quite combinatorial point of view (see [Joan-Arinyo et al. 2003; Hoffmann et al. 2004]); maybe the user intentions should deserve more considerations.

7 Conclusion

This article posed several important problems for GCS and proposed several research tracks: the use of the simplicial Bernstein base to reduce the wrapping effect, the computation of the dimension of the solution set, the pitfalls of graph based decomposition methods, the alternative provided by linear algebra, the witness configuration method which overcomes the limitations of DoF counting and which is even able to probabilistically detect and prove incidence theorems (Desargues, Pappus, Beltrami, hexamy, harmonic conjugate, etc and their duals), the study of incidence constraints, the search for intrinsic (coordinate-free) formulations. Maybe the more surprising conclusion concerns the importance of incidence constraints.

Acknowledgements to our colleagues from the French Group working on geometric constraints and declarative modelling, partly funded by CNRS in 2003–2005.

References

AIT-AOUDIA, S., JEGOU, R., AND MICHELUCCI, D. 1993. Reduction of constraint systems. In *Compugraphic*, 83–92.

AUTODESK. 2005. *DXF reference*, July.

BARNSLEY, M. 1998. *Fractal everywhere*. Academic Press.

BELLEMAIN, F. 1992. *Conception, réalisation et expérimentation d'un logiciel d'aide à l'enseignement de la géométrie : Cabri-géomètre*. PhD thesis, Université Joseph Fourier - Grenoble 1.

BONIN, J. 2002. *Introduction to matroid theory*. The George Washington University.

CHOU, S.-C., SCHELTER, W., AND YANG, J.-G. 1987. Characteristic sets and grobner bases in geometry theorem proving. In *Computer-Aided geometric reasoning, INRIA Workshop, Vol. I*, 29–56.

CHOU, S.-C. 1988. *Mechanical Geometry theorem Proving*. D. Reidel Publishing Company.

COXETER, H. 1987. *Projective geometry*. Springer-Verlag, Heidelberg.

COXETER, H. 1999. *The beauty of geometry. 12 essays*. Dover publications.

CRISTIANINI, N., AND SHAWE-TAYLOR, J. 2000. *An introduction to support vector machines*. Cambridge University Press.

DUFOURD, J.-F., MATHIS, P., AND SCHRECK, P. 1997. Formal resolution of geometrical constraint systems by assembling. *Proceedings of the 4th ACM Solid Modeling conf.*, 271–284.

DUFOURD, J.-F., MATHIS, P., AND SCHRECK, P. 1998. Geometric construction by assembling solved subfigures. *Artificial Intelligence Journal 99(1)*, 73–119.

DURAND, C., AND HOFFMANN, C. 2000. A systematic framework for solving geometric constraints analytically. *Journal on Symbolic Computation 30*, 493–520.

DURAND, C. B. 1998. *Symbolic and Numerical Techniques for Constraint Solving*. PhD thesis, Purdue University.

FARIN, G. 1988. *Curves and Surfaces for Computer Aided Geometric Design — a Practical Guide*. Academic Press, Boston, MA, ch. Bézier Triangles, 321–351.

FOUFOU, S., MICHELUCCI, D., AND JURZAK, J.-P. 2005. Numerical decomposition of constraints. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, ACM Press, New York, USA, 143–151.

GAO, X.-S., AND ZHANG, G. 2003. Geometric constraint solving via c-tree decomposition. In *ACM Solid Modelling*, ACM Press, New York, 45–55.

GAO, X.-S., HOFFMANN, C., AND YANG, W. 2004. Solving spatial basic geometric constraint configurations with locus intersection. *Computer Aided Design 36*, 2, 111–122.

GARLOFF, J., AND SMITH, A. 2001. Solution of systems of polynomial equations by using bernstein expansion. *Symbolic Algebraic Methods and Verification Methods*, 87–97.

GARLOFF, J., AND SMITH, A. P. 2001. Investigation of a subdivision based algorithm for solving systems of polynomial equations. *Journal of nonlinear analysis : Series A Theory and Methods 47*, 1, 167–178.

GOGUEN, J. 1987. Modular specification of some basic geometric constructions. In *Artificial Intelligence*, vol. 37 of *Special Issue on Computational Computer Geometry*, 123–153.

HENDRICKSON, B. 1992. Conditions for unique realizations. *SIAM J. Computing 21*, 1, 65–84.

HILBERT, D. 1971. *Les fondements de la géométrie, a french translation of Grunlagen de Geometrie, with discussions by P. Rossier*. Dunod.

HOFFMANN, C., AND JOAN-ARINYO, R. 2002. *Handbook of Computer Aided Geometric Design*. North-Holland, Amsterdam, ch. Parametric Modeling, 519–542.

HOFFMANN, C., LOMONOSOV, A., AND SITHARAM, M. 2001. Decomposition plans for feometric constraint systems, part i : Performance measures for cad. *J. Symbolic Computation 31*, 367–408.

HOFFMANN, C., SITHARAM, M., AND YUAN, B. 2004. Making constraint solvers more usable: overconstraint problem. *Computer-Aided Design 36*, 2, 377–399.

HU, C.-Y., PATRIKALAKIS, N., AND YE, X. 1996. Robust Interval Solid Modelling. Part 1: Representations. Part 2: Boundary Evaluation. *CAD 28*, 10, 807–817, 819–830.

J. GRAVER, B. SERVATIUS, H. S. 1993. *Combinatorial Rigidity. Graduate Studies in Mathematics*. American Mathematical Society.

JOAN-ARINYO, R., SOTO-RIERA, A., VILA-MARTA, S., AND VILAPLANA, J. 2003. Transforming an unerconstrained geometric constraint problem into a wellconstrained one. In *Eight Symposium on Solid Modeling and Applications*, ACM Press, Seattle (WA) USA, G. Elber and V.Shapiro, Eds., 33–44.

KORTENKAMP, U. 1999. *Foundations of Dynamic Geometry*. PhD thesis, Swiss Federal Institue of Technology, Zurich.

LAMURE, H., AND MICHELUCCI, D. 1998. Qualitative study of geometric constraints. In *Geometric Constraint Solving and Applications*, Springer-Verlag.

LAURENT, M. 2001. Matrix completion problems. *The Encyclopedia of Optimization 3*, Interior - M, 221–229.

LESAGE, D., LÉON, J.-C., AND SERRÉ, P. 2000. A declarative approach to a 2d variational modeler. In *IDMME'00*.

MANDELBROT, B. 1982. *The Fractal Geometry of Nature*. W.H. Freeman and Company, New York.

MICHELUCCI, D., AND FAUDOT, D. 2005. A reliable curves tracing method. *IJCSNS 5*, 10.

MICHELUCCI, D., AND FOUFOU, S. 2004. Using Cayley Menger determinants. In *Proceedings of the 2004 ACM symposium on Solid modeling*, 285–290.

MICHELUCCI, D., AND FOUFOU, S. 2006. Geometric constraint solving: the witness configuration method. *To appear in Computer Aided Design, Elsevier*.

MICHELUCCI, D., AND SCHRECK, P. 2004. Detecting induced incidences in the projective plane. In *isiCAD Workshop*.

MORIN, G. 2001. *Analytic functions in Computer Aided Geometric Design*. PhD thesis, Rice university, Houston, Texas.

MOURRAIN, B., ROUILLIER, F., AND ROY, M.-F. 2004. Bernstein's basis and real root isolation. Tech. Rep. 5149, INRIA Rocquencourt.

NATARAJ, P. S. V., AND KOTECHA, K. 2004. Global optimization with higher order inclusion function forms part 1: A combined taylor-bernstein form. *Reliable Computing 10*, 1, 27–44.

NEUMAIER, A. Cambridge, 2001. *Introduction to Numerical Analysis*. Cambridge Univ. Press.

OWEN, J. 1991. Algebraic solution for geometry from dimensional constraints. In *Proc. of the Symp. on Solid Modeling Foundations and CAD/CAM Applications*, 397–407.

PORTA, J. M., THOMAS, F., ROS, L., AND TORRAS, C. 2003. A branch and prune algorithm for solving systems of distance constraints. In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*.

RICHTER-GEBERT, J. 1996. *Realization Spaces of Polytopes*. Lecture Notes in Mathematics 1643, Springer.

SCHRAMM, E., AND SCHRECK, P. 2003. Solving geometric constraints invariant modulo the similarity group. In *International Workshop on Computer Graphics and Geometric Modeling, CGGM'2003*, Springer-Verlag, Montréal, LNCS Series.

SCHRECK, P. 2001. Robustness in cad geometric constructions. In *IEEE Computer Society Press*, P. of the 5th International Conference IV in London, Ed., 111–116.

SERRÉ, P., CLÉMENT, A., AND RIVIÈRE, A. 1999. *Global consistency of dimensioning and tolerancing*. ISBN 0-7923-5654-3. Kluwer Academic Publishers, March, 1–26.

SERRÉ, P., CLÉMENT, A., AND RIVIÈRE, A. 2002. Formal definition of tolerancing in CAD and metrology. In *Integrated Design an Manufacturing in Mechanical Engineering. Proc. Third IDMME Conference, Montreal, Canada, May 2000*, Kluwer Academic Publishers, 211–218.

SERRÉ, P., CLÉMENT, A., AND RIVIÈRE, A. 2003. Analysis of functional geometrical specification. In *Geometric Product Specification and Verification : Integration of functionality*. Kluwer Academic Publishers, 115–125.

SERRÉ, P. 2000. *Cohérence de la spécification d'un objet de l'espace euclidien à n dimensions*. PhD thesis, Ecole Centrale de Paris.

SHERBROOKE, E. C., AND PATRIKALAKIS, N. 1993. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design 10*, 5, 379–405.

SITHARAM, M., OUNG, J.-J., ZHOU, Y., AND ARBREE, A. 2006. Geometric constraints within feature hierarchies. *Computer-Aided Design 38*, 2, 22–38.

STURMFELS, B. 1993. *Algorithms in Invariant Theory*. Springer.

WINTZ, J., MATHIS, P., AND SCHRECK, P. 2005. A metalanguage for geometric constraints description. In *CAD Conference (presentation only, available at http://axis.u-strasbg.fr/~schreck/Publis/gcml.pdf)*.

WIRSING, M. 1990. *Handbook of Theoretical Computer Science*. Elsevier Science, ch. Algebraic specification, 677–780.

YANG, L. 2002. Distance coordinates used in geometric constraint solving. In *Proc. 4th Intl. Workshop on Automated Deduction in Geometry*.

ZHANG, G., AND GAO, X.-S. 2006. Spatial geometric constraint solving based on k-connected graph decomposition. *To appear in Symposium on Applied Computing*.